The IceT Framework for MetaComputing

Paul A. Gray

Emory Dept. of Mathematics and Comp. Sci.

e-mail: `gray@mathcs.emory.edu`

March 9, 1997

The IceT Foundation for MetaComputing

Paul A. Gray

Emory University, Atlanta, GA 30322

Research supported in part by Army Research Office grant DAAH04-96-1-0083 U. S. Department of Energy Grant No. DE-FG05-91ER25105, NASA Grant NAG 2-828 and the National Science Foundation Award Nos. ASC-9214149 and ASC-9527186.

Novel Aspects of IceT

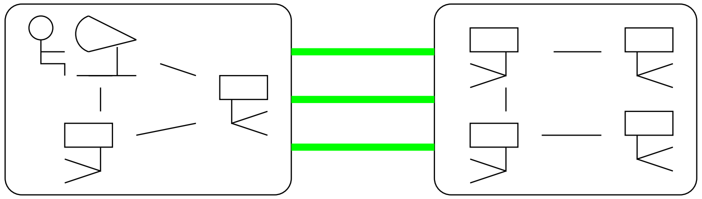
Novel aspects of cluster computing provided for in IceT include:

- "Merging" and "splitting," "plugging" and "unplugging" of multiple users' virtual machines.
- Enhanced portability of processes and data.
- Preservation of traditional distributed programming design.

Consequences of Merging

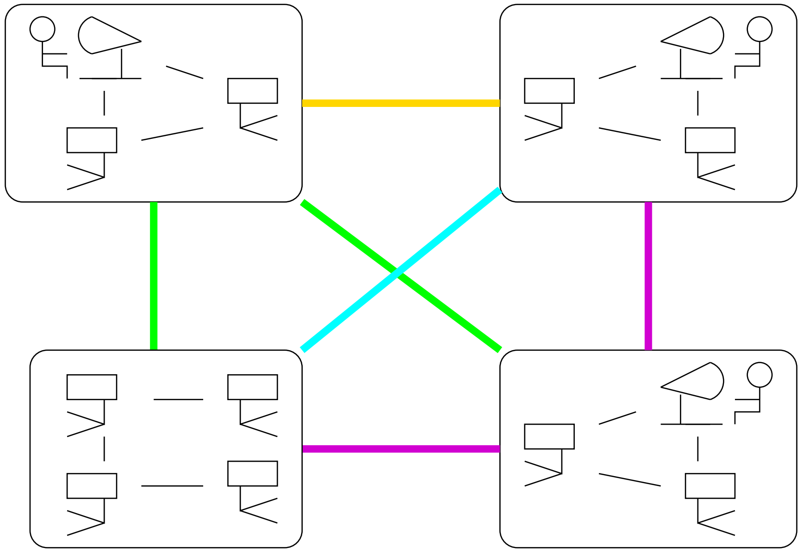
Merging of multiple users' virtual machines leads to a multi-level time-shared virtual machine, making IceT multi-user aware.

- The multi-user awareness leads to better load balancing strategies.
- Enhancement of computational resources to the single user.



Consequences of Merging

- A natural collaborative programming environment.



IceT Components

In order to provide a truly collaborative programming environment user's soft resources must be mutually accessible.

- IceT provides namespace management of geographically distributed *processes*, *data*, *resources* and *users*.
- IceT provides mechanisms to *upload* data *and* processes to remote systems for subsequent manipulation/execution.
  - Perl and shell scripts, Java Bytecode, FORTRAN and C sources, "`.so`" files
  - Dynamic, on-demand uploading $\Longrightarrow$ "Just-in-time–" or "soft–" installations.

Portable Processes

With process fluidity comes additional implications:

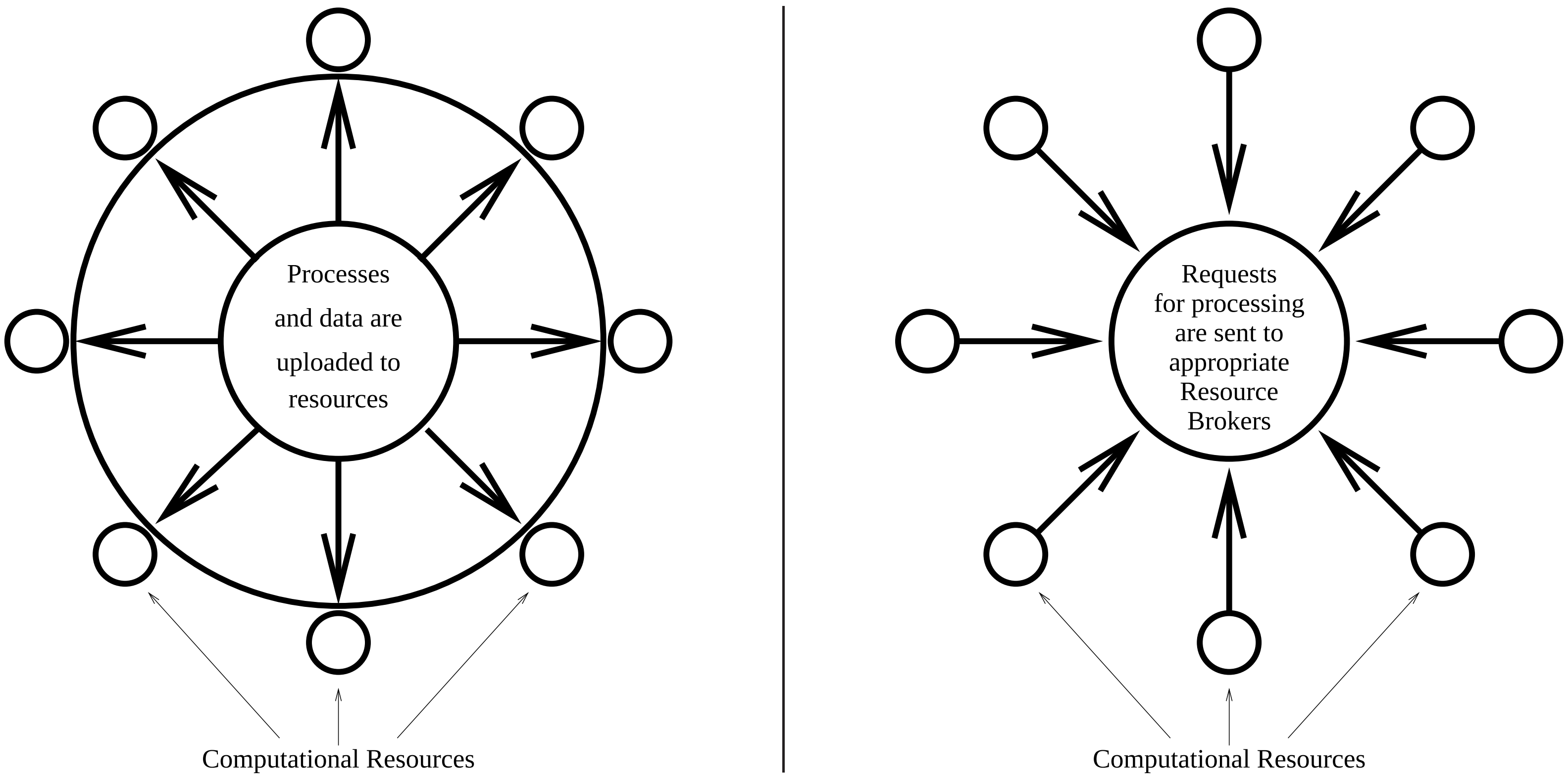
- Sending processes to data makes more sense in some instances.
- Utility to maintain currency of software.
- Powerful collaborative programming tool.

Process and Data "Spoking"

In IceT, processes and data are fluid. Processes may be uploaded to an alternative computational resource for subsequent execution.

- "Spoking" refers to the dissemination of processes and data to computational resources for execution/manipulation.
- Compare spoking to a "Resource Broker" model, where processing requests are sent to centralized locations.

Spoking and Resource Brokerage

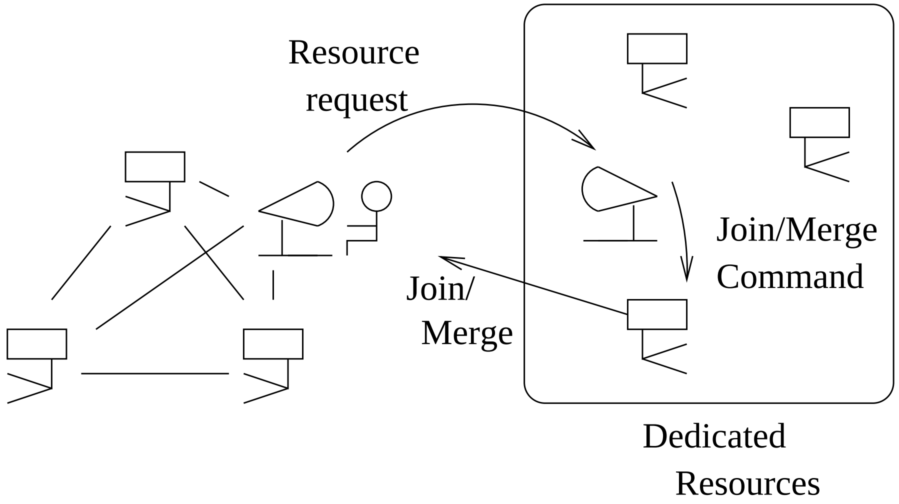


Process "Spoking"

- In the case of Java bytecode, IceT maintains responsibility for porting the main Class file and its dependencies to the remote host.
- IceT has limited facilities for porting "`.so`" files to remote resources for remote linking of system executables.
- IceT will incorporate remote compilation of FORTRAN and C code in the near future.
- Agents — Compubots — ?

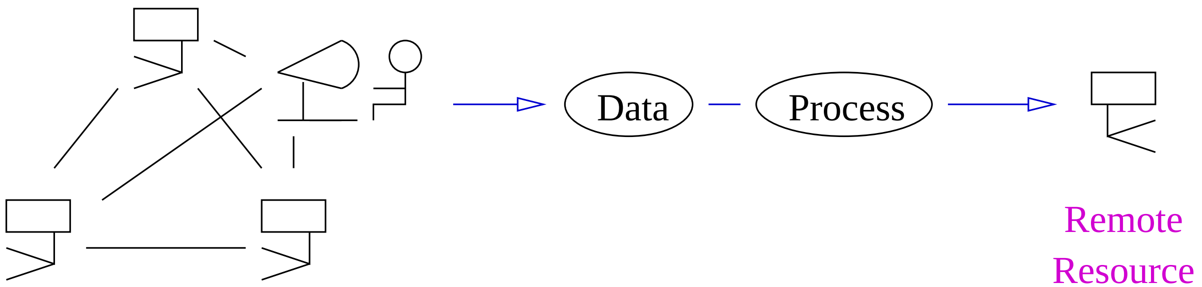
Spoking and Distributed computing

Users saturating their local resources may query a white page server for additional resources. IceT establishes the communication link and manages the details of maintaining the message-passing link with the remote resource.

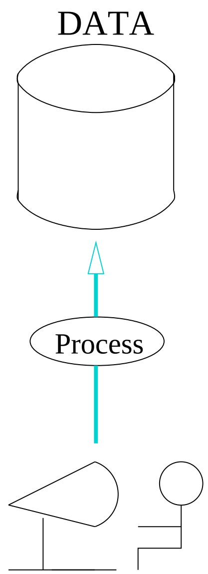


Spoking and Distributed computing

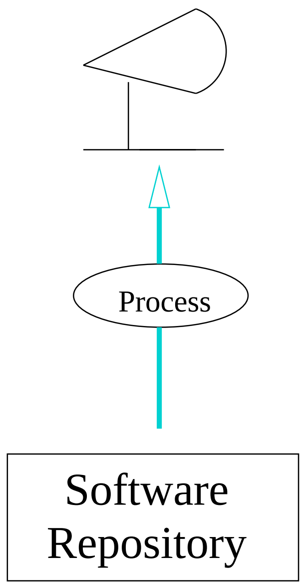
Once IceT has brought the remote resource "on-line," processes and data can be uploaded and executed. Utilization of the remote resource can range from liberal to restrictive, depending upon the remote owner's configuration of IceT.



Other Examples:



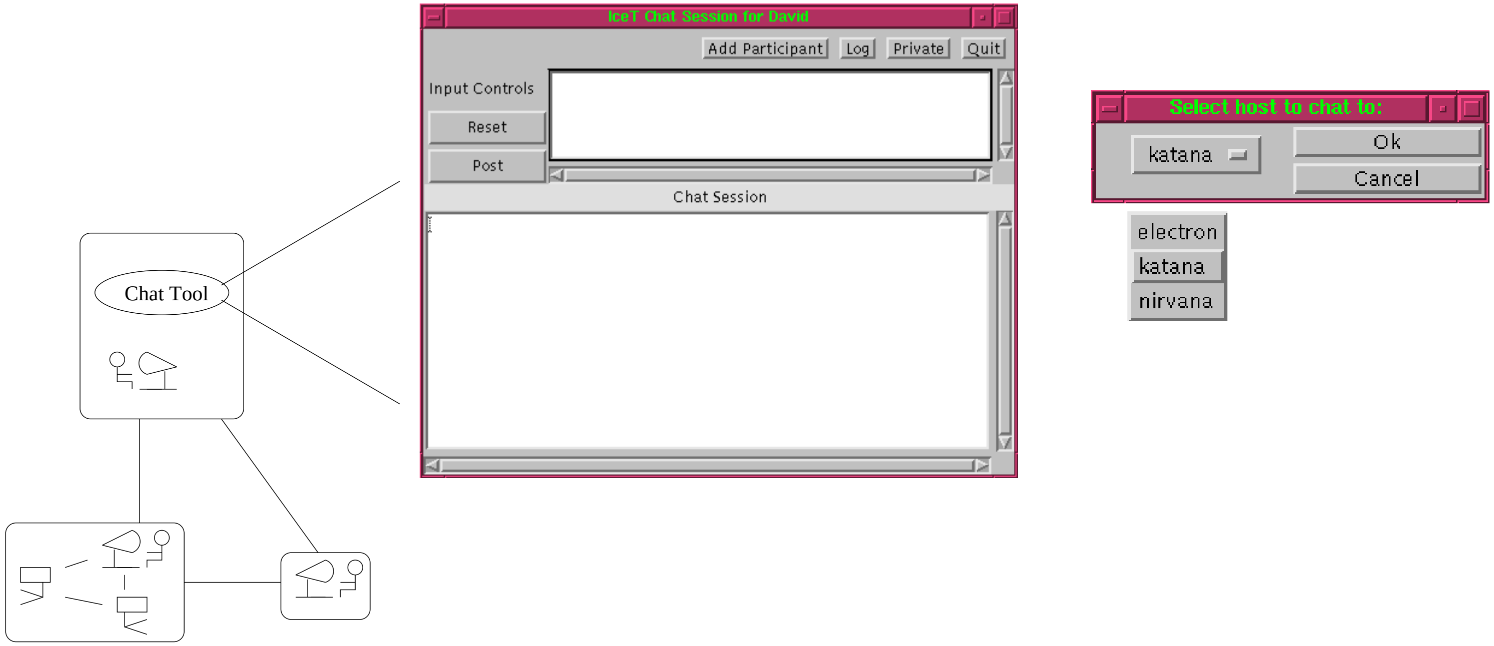
Data Mining



Just-in-Time Installations

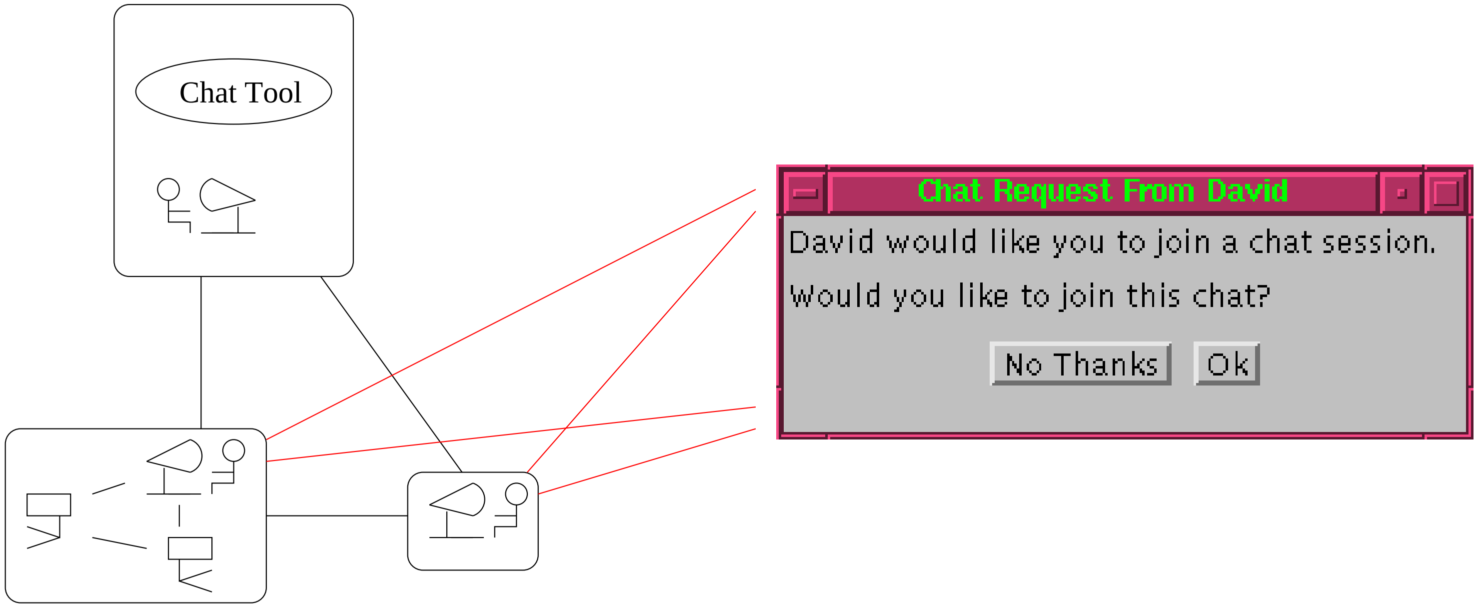
Collaborative Computing Example

The IceTChat tool is an example of a portable process:



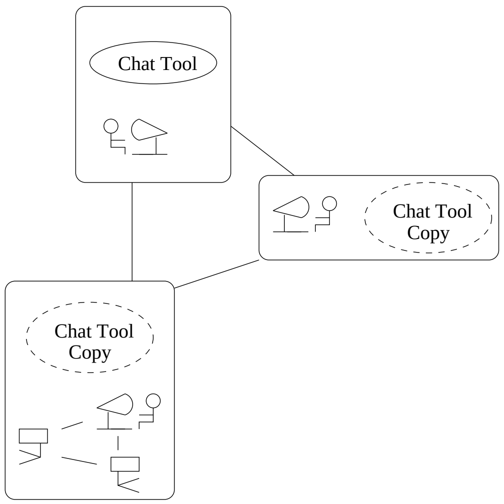
Collaborative Computing Example

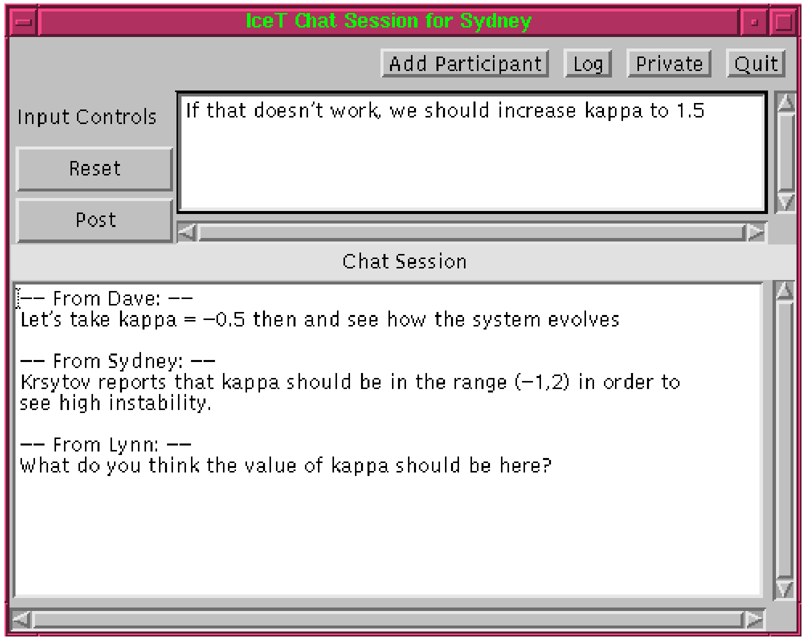
IceT uploads the main process and its dependencies to the remote locations and starts up the process.



Collaborative Computing Example

Once all of the required dependencies are ported to the remote resource, IceT users may interact and collaborate through the tool.





Why Java?

- Standard representations over all platforms of primitives via JVM — a least common denominator.
- Interpreted language → portability of code.
- Facilities for remote instantiation of code for execution via the `ClassLoader` class.
- The Serializable Interface, Remote Method Invocation specification, Java Native Interface (JNI), Java ARchive (JAR), etc: All language-level aspects ammenable to distributed, portable computing.
- Security.

Detrimental Aspects of Java and How They Are Addressed

- Slow!?
  - Just-in-time compilers
  - Uploading, remote linkage and remote execution of native code using "`.so`" files.
  - Uploading and remote compilation of FORTRAN and C source code.
  - IceT Interface to PVM engine via PVM3.4's *Message Box* and JNI PVM calls.

- Legacy Codes
  - Java translations (IBID!)
  - Porting of object code
  - Java front ends to legacy codes.
- Immature, unstable and dynamic state of Java

Performance

- Benchmarks for computational tasks exhibiting high-granularity show IceT "comparable" to the equivalent PVM code written in C.
- IceT/Java times are about $1.25\times$ – $4\times$ that of the PVM/C times when using Just-in-time compilers.
- Improvements expected as Java and IceT mature.

Performance Benchmarks

The following are performance times for the computation of multiplying two 400x400 matrices:

# processes computations were distributed over:	Java-Based Compute Engine supported by an IceT Environment	C-based Compute Engine supported by an IceT Environment	C-based Compute Engine supported by a PVM3.3 Environment
1	291.796	41.480	37.134
2	161.589	45.116	24.608
4	87.044	28.920	16.575
5	72.927	26.354	16.506
8	50.695	21.287	16.272

Distributed Computing Aspects

- Configurability of the virtual machine
  - add and delete hosts, merging virtual machines
- Interaction with the multi-level virtual machine
  - spawning of local processes on local and remote resources
  - local access to remote data and processes
- Traditional aspects of distributed computing as in PVM/MPI
- Portability of data and processes